Under consideration for publication in Knowledge and Information Systems

Hybrid Reasoning Technique for Improving Context-Aware Applications

Matthias Strobbe, Olivier Van Laere, Bart Dhoedt,
Filip De Turck and Piet Demeester
Ghent University - IBBT, Department of Information Technology,
Gaston Crommenlaan 8, bus 201, 9050 Ghent, Belgium

Abstract. With the rapid adoption of GPS enabled smart phones and the fact that users are almost permanently connected to the Internet, an evolution is observed towards applications and services that adapt themselves using the user's context, a.o. taking into account location information. To facilitate the development of such new intelligent applications, new enabling platforms are needed to collect, distribute and exchange context information. An important aspect of such platforms is the derivation of new, high-level knowledge by combining different types of context information using reasoning techniques. In this paper we present a new approach to derive context information by combining case-based and rule-based reasoning. Two use cases are detailed where both reasoners are used to derive extra useful information. For the desk sharing office use case, the combination of rule-based and case-based reasoning allows to automatically learn typical trajectories of a user and improve localization on such trajects with 42%. In both use cases, the hybrid approach is shown to provide a significant improvement.

Keywords: Case-Based Reasoning, Rule-Based Reasoning, Context-Aware Services, Location-Based Services, User Profiling

1. Introduction

Today, location-based and context-based services are well established. They turn our homes, offices, cars, cities, etc. into smart environments where highly adaptive services are dynamically deployed, updated or replaced. An example is a context-based city guide which gives information about interesting locations in

Received Jul 16, 2010
Revised Mar 11, 2011
Accepted Apr 02, 2011

the neighbourhood of the user taking into account the user's preferences, the time of the day, the weather, etc. Important catalysts for the use of context-aware applications are the rapid spread of GPS enabled smart phones and accompanying application stores (such as the currently popular iPhone and App Store) and the fact that users are almost permanently connected to the Internet. The market of location and context based services is expected to expand significantly the next years (Gartner, 2009).

However, the incompleteness and/or inaccurateness of context information is still a major problem. There might also arise inconsistenties when several context sources deliver the same kind of information, for example a system that uses GPS signals for outdoor localization and WiFi signals for indoor localization. Developing context-aware services is labor-intensive in terms of aggregating and combining different kinds of context information to high-level knowledge. Facilitating the development, deployment and management of these services requires new context-aware service frameworks. An important component of these frameworks will be an enabling service dealing with the collection, distribution and exchange of context and location information.

To address these challenges we developed CASP, a Context-Aware Service Platform that takes care of the aggregation and abstraction of context information using ontologies for representing the information. CASP collects context information in a certain domain, facilitating the development of new services in that domain. CASP has already been used in diverse application areas ((Strobbe et al., 2007), (Ongenae et al., 2008), (Strobbe et al., 2010)), and is described in more detail in section 3Architecturesection.3.

To facilitate hybrid reasoning, i.e. exploit intelligent combinations of rule-based and case-based reasoning, CASP has been extended to host different reasoners. More specifically, a case-based reasoner was added to the system, such that high-level context information can be derived using rules, cases or a combination of both. The reason for adding a case-based reasoner to CASP is that rule-based reasoning and case-based reasoning are complementary when it comes to distilling new knowledge. Rules represent general knowledge of the domain, whereas cases capture specific knowledge. Rule-based systems solve problems from scratch, while case-based systems use pre-stored situations to deal with similar new instances. Therefore, the combination of both approaches turns out to be natural and useful (Prentzas and Hatzilygeroudis, 2007).

Case-based reasoning is based on the principle of using past experiences to take a decision in a current situation. A case-based reasoning system is typically made up of four steps (Aamodt and Plaza, 1994):

1. Retrieve: In this step the most similar case or cases to a certain problem are retrieved.
2. Reuse: The solutions of the retrieved cases are used to solve the problem at hand.
3. Revise: The proposed solution is revised based on feedback.
4. Retain: The revised case or a new case is stored in the case base to be used for future problem solving.

Figure 1Overview of the CBR cycle by Aamodt and Plaza (Aamodt and Plaza, 1994) consisting of 4 steps: retrieve, reuse, revise and retainfigure.1 illustrates this cycle. An initial description of a problem defines a query. This new case is used to *retrieve* a case from the collection of previous cases. The retrieved

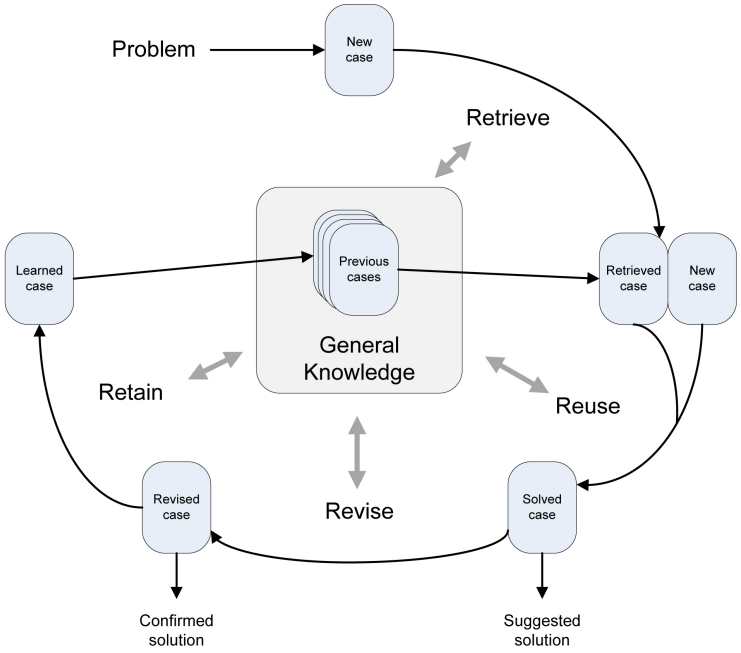


Fig. 1. Overview of the CBR cycle by Aamodt and Plaza (Aamodt and Plaza, 1994) consisting of 4 steps: retrieve, reuse, revise and retain.

case is combined with the new case - through *reuse* - into a solved case, i.e. a proposed solution to the initial problem. Through the *revise* process this solution is tested for succes and repaired if failed. During the *retain* process, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases. General knowledge usually plays a part in this cycle, by supporting the CBR steps.

The remainder of this paper is organized as follows: section 2Related Worksection.2 discusses related work. In section 3Architecturesection.3 we present the architecture of CASP with the newly added case-based reasoner. Two use cases are detailed in sections 4Use Case: Desk Sharing Office Environmentsection.4 and 5Use Case: Enhanced Instant Messaging Communication Servicesection.5 respectively where both reasoners are combined to derive extra useful information. Section 6Conclusions and Future Worksection.6 states the conclusions and discusses future work.

2. Related Work

2.1. Reasoning Techniques

In the literature a lot of different types of knowledge-based reasoning exist. An often used technique is **Rule-Based Reasoning** where domain specific information is offered to applications by means of rules, which improves the reusability of the application. The rules are a translation of the knowledge of a domain expert

and typically have a *"IF condition THEN action"* representation. Rule-based reasoning requires that general knowledge about a certain domain is available and can be expressed by rules. Examples of well-known rule-based reasoners are Jess (Jess, 2008) and Drools (JBoss, 2010). Semantic reasoners such as Pellet (Clark & Parsia, 2010) and RacerPro (Racer, 2010) also support rules. **Case-Based Reasoning** is based on the intuition that problems tend to recur, meaning that new problems are often similar to previously encountered problems and, as a consequence, that past solutions may be of use in the current situation (Leake, 1996). Case-based reasoning is particularly applicable to problems where earlier cases are available, even when the domain is not understood well enough for deep domain modeling (Garca, 2008). **Diagrammatic Reasoning** uses visual representations for the reasoning. Examples of such visual representations are charts, graphs, maps, etc. This kind of reasoning requires that the problem domain can be represented using a diagram (Glasgow et al., 1995). With **Constraint Based Reasoning** a problem is expressed as a number of variables and constraints for those variables (Apt, 2003). Constraint based reasoning is similar to Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP). **Model Based Reasoning** uses a model of a problem domain that consists of smaller blocks (like the pumps and valves in a hydraulic system) to reason about a problem (Magnani and Nersessian, 2002). **Qualitative Reasoning** systems reason about the behavior of physical systems, without having precise quantitave information. Observing pouring rain and a river's steadily rising water level is sufficient to make a prudent person take measures against possible flooding - without knowing the exact water level, the rate of change, or the time the river might flood (Iwasaki, 1997). **Fuzzy Reasoning** techniques take into account that not all information is known at every moment in time or that variables might have more than one value (Novk et al., 1999). **Logic Reasoning** uses techniques from logic to abstract a problem and reasons about the problem using the rules of a formal language. Afterwards the results need to be interpreted. Semantic reasoners such as Pellet (Clark & Parsia, 2010), FaCT++ (Fact++, 2009) and RacerPro (Racer, 2010) use description logics to perform reasoning.

Context-aware systems often use rule-based reasoning and logic reasoning. For example the Context Awareness in Sensing Environments framework (CASE) (Chong et al., 2011) uses rules to improve sensor energy consumption in wireless sensor networks and the CAS-Mine framework (Baralis et al., 2010) extracts rules from context data allowing service providers to personalize their services based on the current user context. In (Bouamrane et al., 2010) logical inference on ontologies is used to generate a personalized preoperative assessment report for patients in hospitals. These kinds of reasoning techniques are especially interesting if general knowledge about the domain is known or can be derived. Besides these reasoning techniques, case-based reasoning and fuzzy reasoning seem very interesting to be used in context-aware frameworks. With case-based reasoning historical context information can be exploited by learning trends and habits. In (Kofod-Petersen and Aamodt, 2006), knowledge intensive case-based reasoning is used to create an ambient intelligent system. (De Paz et al., 2010) uses case-based reasoning for monitoring the CO_2 exchange rate between the atmosphere and oceans. In (Leake et al., 2006), an overview is given of the possibilities of case-based reasoning in a home environment. Case-based reasoning is suitable for such an environment as decisions can be explained and the learning process can

be easily adjusted. A system that is understandable by the user will be perceived as more reliable by the user.

Fuzzy reasoning is interesting as context-aware systems often contain unprecise or even contradictory information. In (Mylonas et al., 2008) fuzzy representations of user interests, user context and content meaning are combined with ontologies to improve the accuracy and reliability of personalized information retrieval. In (Mntyjrvi and Seppnen, 2003) applications in a handheld device are adapted according to multiple fuzzy contexts. Fuzzy controllers use a fuzzy rule base to adjust the applications using membership functions of several context features.

In the domain of context information there are few systems that combine several types of knowledge-based reasoning. In this paper we investigate the possibilities of the combination of rule-based reasoning and case-based reasoning.

2.2. Context Frameworks

In recent years a number of context platforms have been developed to relieve application developers from the aggregation and abstraction of location and context information, and the derivation of high-level contexts. One of the first platforms was the Context Toolkit (Dey, 2000), a Java framework allowing the rapid prototyping of sensor based context-aware applications. However, the Context Toolkit does not focus on a context model for exchanging information. In recent years a number of context platforms have been developed using formal context models based on ontologies ((Gu et al., 2005), (Ejigu et al., 2008), (Ko et al., 2007), (rong Jih et al., 2008)). Using ontologies provides semantic uniformity and interchangeability of context information in a heterogeneous setting such as pervasive computing environments. Our platform CASP belongs to this latter series of platforms. Most of the above cited platforms use reasoners based on rules and description logics to derive high-level information.

Some research has already been done on systems that combine rule-based reasoning and case-based reasoning. In (Kumar et al., 2009) both reasoning techniques are combined for a clinical decision support system and (Prentzas and Hatzilygeroudis, 2007) mentions the legal and medicine domains as popular application fields for integrating rules and cases. But in the domain of context information, research which combines rule-based and case-based reasoning is still very limited. To fill this gap we extended out platform CASP with case-based reasoning which gives it an added value in comparison with the mentioned context frameworks.

3. Architecture

3.1. CASP

Figure 2Overview of the overall CASP architecturefigure.2 gives an overview of the layered architecture of the Context-Aware Service Platform (CASP (Strobbe et al., 2006), (Strobbe et al., 2007)).

At the bottom, the *persistence layer* ensures the persistence of context information. This comprises static context information about users, devices, the environment, etc. More dynamic information, such as positions of users, can also

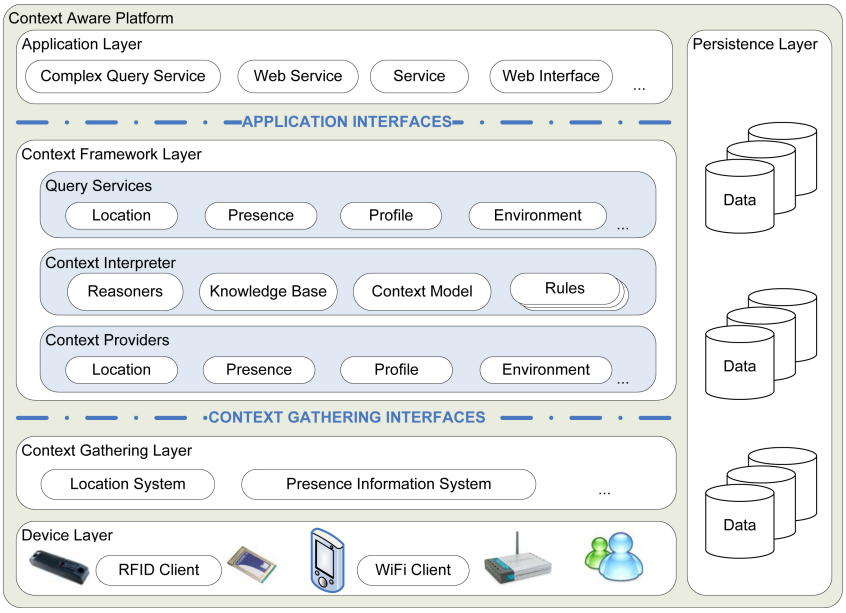


Fig. 2. Overview of the overall CASP architecture

be stored. In this way, the history of context information can be exploited and trends or habits can be derived.

The *device layer* includes all devices (and software components on those devices) that deliver context information. For example, a WiFi client on a PDA can measure the received signal strengths of access points in the surroundings and send this information to the location system.

The *context gathering layer* takes care of the acquisition of specific context information, for instance location or profile info. To improve the modularity of the platform, context gathering interfaces are defined between the context gathering layer and the context framework layer for important types of context information such as location, presence, profiles, etc. It is sufficient to implement such an interface to add a context gathering component to the platform.

The *context framework layer* is responsible for the aggregation of the context information according to a formal context model and the derivation of implicit information by reasoning. Context information coming from the context gathering layer is translated to ontologies by the context providers and gathered in a knowledge base. All context providers implement a common interface making them easily pluggable into the platform. Derivation of high-level knowledge is done by the reasoner components. The reasoners can also be used for validation purposes. For example, if several context providers deliver the same kind of information, there will probably be inconsistencies from time to time. Based on reliability and accuracy parameters, a decision can be made on the correctness of the information. The query services enable and facilitate the retrieval of context information. They translate ontology constructs to objects and expose an application interface towards the services. This relieves application developers from writing error-prone queries and translating the results to objects themselves.

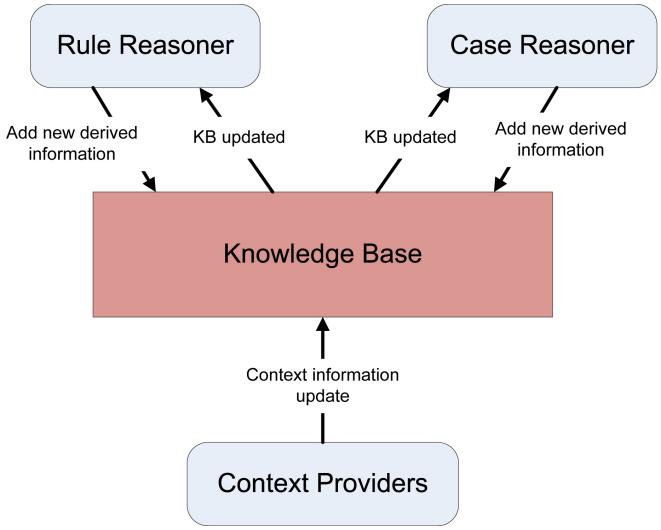


Fig. 3. Detailed architecture of the core of CASP showing the interactions between context providers, knowledge base and the rule-based and case-based reasoners.

The *application layer* contains interfaces and web services that permit easy querying of the knowledge base and applications that use the context information.

3.2. CASP Improvements

In this paper we present a new version of CASP which contains an additional case-based reasoner. The previous version of CASP only contained a rule-based reasoner which is useful for deriving general knowledge in a certain domain. With the addition of a case-based reasoner, our platform is now able to also learn specific knowledge and exploit historical information. The presence of two complementary reasoners allows hybrid reasoning, i.e. intelligent combinations of both rule-based and case-based reasoning to resolve inconsistencies, deal with unprecise information and derive new, high-level information. To the extent of our knowledge, CASP is one of the first platforms in the domain of context-aware computing, that contains both rule-based and case-based reasoners.

Figure 3Detailed architecture of the core of CASP showing the interactions between context providers, knowledge base and the rule-based and case-based reasonersfigure.3 show the architecture of the new core of the CASP platform. When a context provider updates information in the knowledge base, the rule-based and case-based reasoners are notified. The rule-based reasoner will evaluate the provided rules and add the derived information to the knowledge base. The case-based reasoner will retrieve similar situations and use the corresponding solutions to update the information in the knowledge base.

Figure 4Detailed architecture of the case-based reasonerfigure.4 shows in more detail the architecture of the case-based reasoner. It consists of three main components. The *Case Solver* will react on knowledge base updates and will retrieve matching case(s) from the *Case Base* for the current situation. When

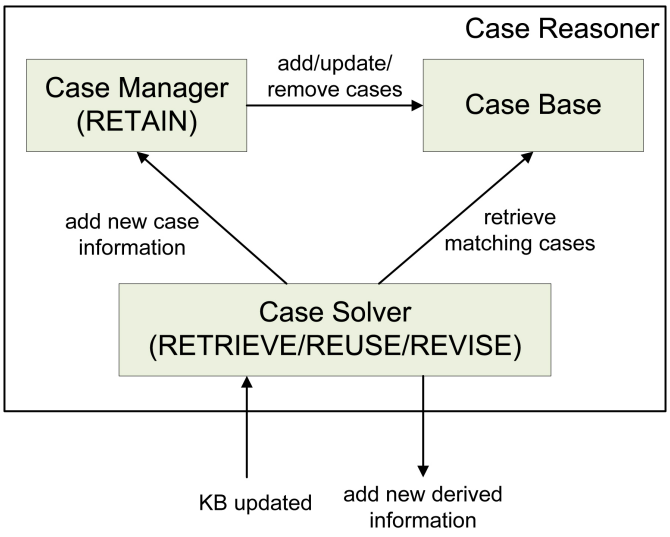


Fig. 4. Detailed architecture of the case-based reasoner.

a well-matching case is found the *Case Solver* will use that case to update the context information in the knowledge base. The *Case Solver* might be notified by the knowledge base of problems with the proposed solution and consequently, the case at hand can be revised.

The *Case Solver* will possibly notify the *Case Manager* about a new case (for example when no well-matching case is found or when the used case is updated during the revise step). The *Case Manager* will use these new and adapted cases to update the *Case Base*. By providing information about new cases to the *Case Manager* instead of directly storing that information in the *Case Base* allows those cases to be filtered or aggregated before they are stored. This might be useful if the case-based reasoner is for example used to derive trends and habits and the system administrator doesn't want the *Case Base* to be overloaded with every single occurred situation. This will be illustrated with the two use cases presented in sections 4Use Case: Desk Sharing Office Environmentsection.4 and 5Use Case: Enhanced Instant Messaging Communication Servicesection.5.

3.3. Implementation Details

Each layer of CASP is designed in a modular way with a limited number of dependencies using the OSGi platform (The Open Services Gateway Initiative (OSGi), 2011) as a basis. If a context gathering component is added or removed from the context gathering layer, the context provider in the context framework layer will detect this and the context information in the knowledge base will be updated. Similarly, when a context provider is removed from the context framework layer, the context gathering components will notice that there is no suitable context provider available and no information will be exchanged. This way, components can be added, started, updated, removed or stopped while the other components of the platform keep running. The modular design allows a

deployment in a distributed manner. If the load increases, the databases can be deployed on several servers, the context gathering components can be distributed across multiple servers and the knowledge base can be deployed in a distributed manner.

To describe the context models OWL (Web Ontology Language) (W3C, 2004) is used. OWL is an ontology language proposed by W3C, as a vocabulary extension of RDF. OWL allows automated processing of terms and relationships between terms in vocabularies, by representing the meaning of those terms. Domain knowledge can be accurately described by means of classification, modeling dependencies and restrictions on these dependencies. Other ontologies can be imported, encouraging reuse.

For the implementation of the knowledge base and rule-based reasoner the Jena2 Semantic Web Toolkit (Jena 2 Semantic Web Toolkit, 2010) is used. This Java library offers an OWL API and a rule-based inference engine.

For the implementation of the case base and case-based reasoner the jCOLIBRI Case-Based Reasoning Framework (GAIA - Group for Artificial Intelligence Applications, 2010) is used. This Java library offers a complete CBR development architecture, including mechanisms to retrieve, reuse, revise and retain cases and is designed to be easily extended with new components. We also provided a number of abstract classes to assist the developer in retrieving and retaining cases.

4. Use Case: Desk Sharing Office Environment

4.1. Motivation

This use case was developed for the premises of a large company in Belgium. After moving to a new location desk sharing was introduced. In a desk sharing environment, employees no longer have a dedicated desk anymore. When they arrive at work, they can choose whatever desk available in the building to start working at. In such an environment, several problems arise. Visitors who have an appointment with an employee do not know where to find the employee. Even an employee who needs a colleague for a meeting is not always able to locate him or her.

The desk sharing office use case is a web application consisting of several services. The application allows employees or visitors to locate other users and to see how someone can be reached. The user has to fill in the name of the contact and as a result the path from his or her current location to the location of the contact, together with the status of that person and some extra information such as email address, phone number, etc. is returned. Figure 5Screenshot of the web application showing information about an employee and the shortest path to that employeefigure.5 presents a view on the graphical front end developed for this use case.

When the person the user is looking for, is not available or not present, a default location can be returned, e.g. the reception desk or a colleague who might be able to help. The application provides the presence status of persons, which allows to see who is present or available. If a person is available, one can click on a telephone icon to setup a phone call to the person through VoIP. A call is then set up between the phones that are located the closest to both persons.

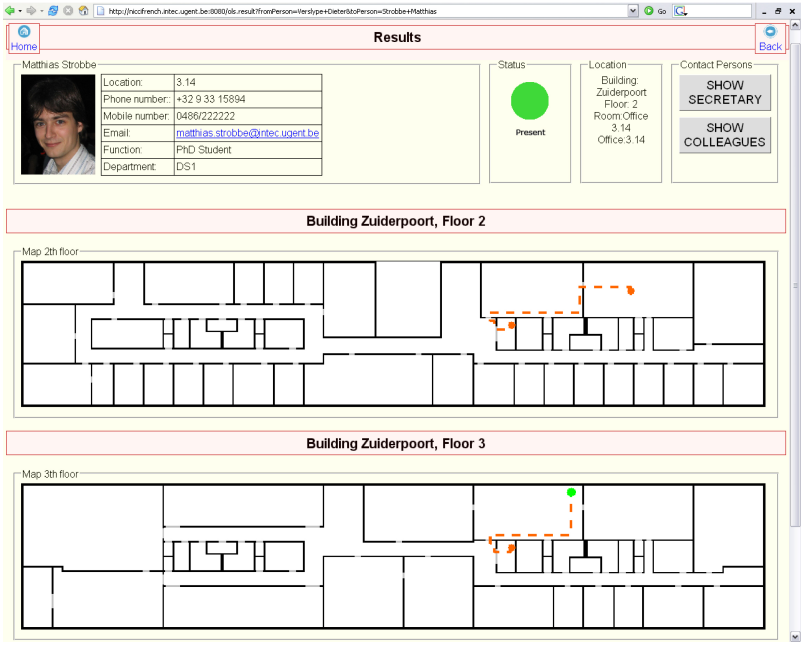


Fig. 5. Screenshot of the web application showing information about an employee and the shortest path to that employee.

4.2. Design Details

An ontology for an office environment has been designed to enable the different services offered by the web application. This ontology is shown in figure 6Designed ontology for the desk sharing office use casefigure.6.

To determine the location of a user, the personal devices of that user are tracked using wired and wireless techniques, making use of the existing network infrastructure to limit costs. As an employee can have several personal devices, and possibly be tracked by different location systems, a person might have several locations. These locations can be different at the same instance of time, e.g. an employee can be walking around the building with his PDA, while his laptop is still on his desk. Moreover, the accuracy of wireless location determination techniques is not perfect. We used the rule-based reasoner of CASP to infer the most probable location of a user. Rules are also used for inferring presence information. When a user has a location, his status automatically becomes online. When he leaves the building, his status is changed to offline. When a user manually changes his location to busy or away, the rules will assign the user a specific default location associated with the status, e.g. a colleague or the reception desk. More details about the used techniques for localization and the rules used for resolving location inconsistencies and deriving user statuses can be found in (Strobbe et al., 2007).

In this paper we use the rule-based reasoner and the new case-based reasoner of CASP to improve the localization of the user. We assume that the underlying location systems supply locations with a certain random error (for example $3m$

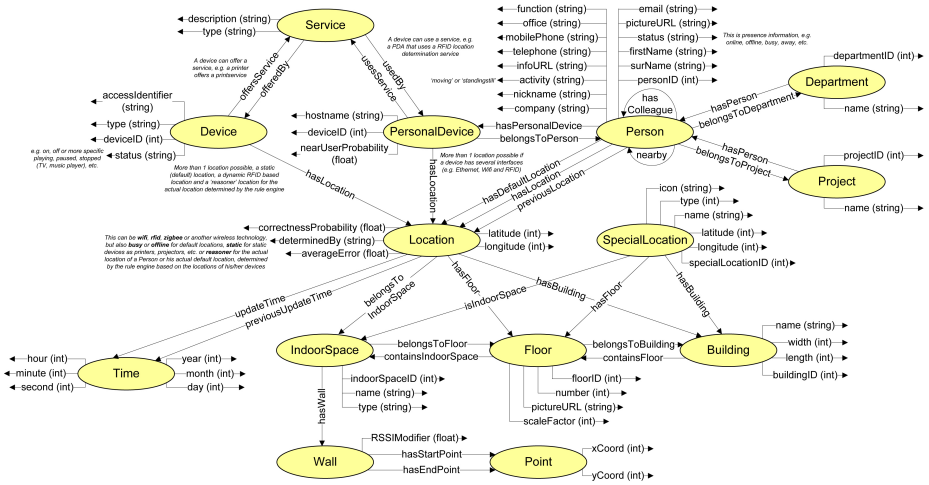


Fig. 6. Designed ontology for the desk sharing office use case.

on average). Using rule-based and case-based reasoning we try to improve the location determination. This is illustrated in pseudocode in Algorithm 1Design Detailsalgocf.1 and discussed in more detail in the following sections where we focus on both reasoning techniques.

4.2.1. Rule-Based Reasoning

When a location for a personal device of the user is added to the knowledge base by a location provider, the *insert_location_person* rule will fire:

```
[insert_location_person:
  (?x rdf:type Person)
  (?x hasPersonalDevice ?y)
  (?y hasLocation ?l)
  -> (?x hasLocation ?l)
  fireRuleEvent('location' ?x ?l)]
```

This rule will set the location of the user to the location of his personal device. At the same time a rule event will be fired that will be intercepted by a component which implements a simple motion model. For this motion model, we use a location model which generates a distribution from the added location and an actual motion model based on the previous location of the user.

To describe the location model and motion model we use the following parameters: l is the location added by the location provider. Surrounding locations are denoted with l_s and are situated at a distance d_s from location l. l_p denotes the previous location of the user, at a distance d_p from a surrounding location of that previous location. The distance a user travels between two location updates (during timespan t) is denoted by d_t. We assume an average user speed v.

The location model (Algorithm 1Design Detailsalgocf.1, line 6) takes into account the error on the supplied location l. Assuming this location l and surrounding locations l_s receive a location probability P_l using a Gaussian distribution:

```
Upon fire of insert_location_person rule for location l
begin
    t_l ← KB // Retrieve location update time
    l_prev ← KB // Retrieve previous location
    t_{l_prev} ← KB // Retrieve previous location update time
    locModel = calculateLocationModel(l)
    motionModel = calculateMotionModel(t_l, l_prev, t_{l_prev})
    combinedModel = calculateCombinedModel(locModel, motionModel)
    bestLocation = selectBestLocation(combinedModel)
    adaptUserLocation(bestLocation)
end
Upon fire of improve_location_in_meeting rule for location l
begin
    bestLocation = calculateClosestLocInMeetingRoom(l, floorplan)
    adaptUserLocation(bestLocation)
end
Upon update of KB for location l of user u
begin
    a ← KB // Retrieve user activity
    t_l ← KB // Retrieve location update time
    if a has changed && a == 'moving' then
        matchingCase = searchMatchingCaseInCaseBase(u, l, t_l)
        if matchingCase != null then
            u.setCurrentCase(matchingCase)
        end
        l_s ← l // Store start location case
        t_{l,s} ← t_l // Store start time case
    end
    if a == 'moving' && u.getCurrentCase() != null then
        case = u.getCurrentCase()
        if useMethod1 == true then
            bestLocation = calculateClosestLocationOnPath(l, case)
        end
        else
            bestLocation = calculateBestLocOnPathCloserToEnd(l, case)
        end
        if bestLocation != null then
            adaptUserLocation(bestLocation)
        end
        else
            u.setCurrentCase(null)
        end
    end
    if a has changed && a == 'standing still' then
        u.setCurrentCase(null)
        caseManager.addNewCaseInfo(u, l_s, l, t_{l,s})
    end
end
```

Algorithm 1: Localization improvements using rule-based and case-based reasoning.

$$P_l(d_s) \sim \exp(-\frac{d_s^2}{\sigma_l^2}) \tag{1}$$

We assume the average error (μ_{error}) on locations added by the location provider is known, for example $3m$ which is a value that can be expected when WiFi is used with commodity hardware inside a building (Elnahrawy et al., 2004). We use this value to determine a value for σ_l^2 and impose the condition that a location at a distance equal to this average error, has a correctness probability $P_l(\mu_{error})$ that is x times smaller than the probability of the returned location l. This gives us the following expression for σ_l^2:

$$\sigma_l^2 = \frac{\mu_{error}^2}{\ln x} \tag{2}$$

For the user motion model (line 7), we use the information defined in the ontology presented in figure 6Designed ontology for the desk sharing office use casefigure.6, more specifically the previous location of the user and the update timestamp for that location. We assume that there is a large probability that a user is still at his previous location or somewhere near to that location. We use again a Gaussian distribution to model this motion probability P_m:

$$P_m(d_p) \sim \exp(-\frac{d_p^2}{\sigma_m^2}) \tag{3}$$

To determine σ_m^2 we need to take care of the time between the measurement of the previous location of the user and the measurement of the current location. The larger this time span, the further a user may have moved and the less reliable the motion model becomes. In case of a large time span the motion model should approximate a uniform distribution, implying a large value for σ_m^2. In our model, we take an average speed value for an average user. In addition, we take the assumption that a location at a distance that can be traveled at this average speed during the measured time span has a probability that is half the probability as if the user would stay at his previous location. As $d_t = v \times t$ we obtain the following expression for σ_m^2:

$$\sigma_m^2 = \frac{v^2 t^2}{\ln 2} \tag{4}$$

To determine the most probable location of a user, a combined probability (P_c) is calculated using the location model and motion model (line 8):

$$P_c(l) = P_l(l) P_m(l) \tag{5}$$

The location of the user is then changed to the location with the highest combined probability (lines 9-10).

Such a motion model is often implemented on a lower level, inside a specific location system (Fox et al., 2003). In contrast, our approach is useful for systems that do not use such algorithms. By improving the locations in the knowledge base, we can also incorporate other context information. To illustrate this, we

implemented a second rule that takes status and environment information into account.

The *improve_location_in_meeting* rule will fire when a user has set his status to 'in_meeting' and is located outside a meeting room:

```
[improve_location_in_meeting:
  (?x rdf:type Person)
  (?x hasLocation ?loc)
  (?loc determinedBy 'improved')
  (?x status 'in_meeting')
  (?loc belongsToIndoorSpace ?i)
  noValue(?i type 'Meeting Room')
  -> fireRuleEvent('inmeetinglocation' ?x ?loc)]
```

The component that intercepts the rule event will adapt the location of the user to the closest location inside a meeting room (Algorithm 1Design Detailsalgocf.1 lines 12-16).

4.2.2. Case-Based Reasoning

Users often have some typical trajectories during a day at the office. For example arriving at 9 AM, getting a coffee around 10 AM, having lunch around 12.30 PM, maybe a daily project follow up meeting at 4 PM, etc. If you know these patterns you can improve localization as you know the path the user will take. Of course these patterns are different for every user and might change over time. The system should not only use the patterns to improve the localization, it should also learn them in an automatic way.

To learn such trajectories one could for example use Markov models, which are often used for modelling traffic characteristics ((Sowunmi et al., 2009), (Stamoulakatos et al., 2007)). We however choose to use a case-based reasoner, as this approach is more generically applicable (i.e. in domains different from localization problems).

The cases in the case base model the trajectories of a user with the time, start point and end point. When a user starts to move, the case base is queried for a matching case (Algorithm 1Design Detailsalgocf.1 lines 21-22). If found, the shortest path between start and end point is used to improve the location of the user (lines 29-43). When a large deviation between the location of the user and the path of the case is observed, the case is no longer used for location improvements (line 41).

To determine if a case is well-matching, the user, time and start point attributes are compared with the situation at hand. For these 3 attributes a match value is calculated and the average value of these match values is compared with a threshold. The calculation of the match value for the user attribute is as follows: 1 for a matching user and 0 otherwise. For the time attribute we calculate the time difference in units of τ minutes (d_τ) and convert this value to a value between 0 and 1 using equation (6Case-Based Reasoningequation.4.6):

$$TimeSimilarity = \frac{1}{d_\tau + 1} \tag{6}$$

Doing so a trajectory started τ minutes earlier or later than a case in the case base will result in a match value of 0.5. If the time difference is three times larger, the match value will lower to 0.25.

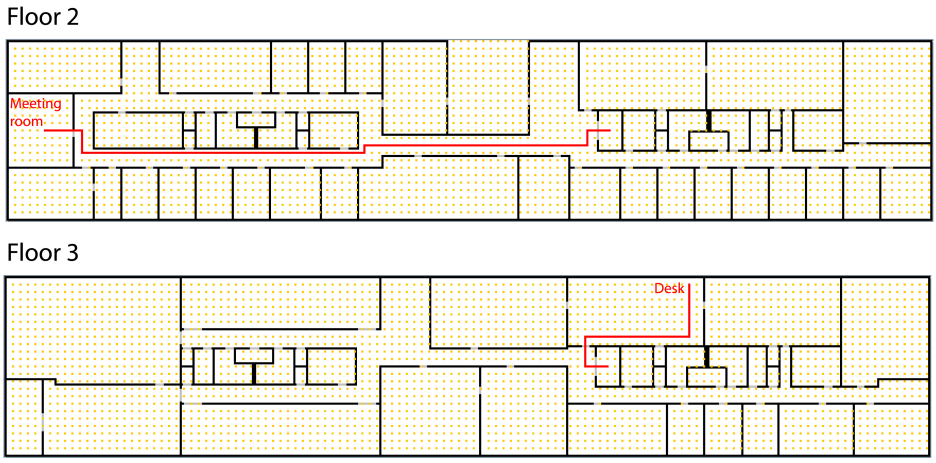


Fig. 7. Maps of the office environment that is used for the simulation. A simulated trajectory from the desk of a employee to a meeting room is shown.

For the start point attribute we do something similar. We calculate the physical distance (in meter) covered by the shortest path between the current location of the user and the start point of the inspected case ($d_{shortestpath}$). This value is converted to a value between 0 and 1 using equation (7Case-Based Reasoningequation.4.7):

$$LocationSimilarity = \frac{1}{d_{shortestpath} + 1} \tag{7}$$

At the same time, the current traject of the user is stored as this forms a possible new case (lines 26-27). The path of the user is monitored and the time, start and end locations are stored in the case manager component. These possible cases also form a case base where similar possible cases are grouped. If a possible case occurs often during a certain period of time, the case is considered as an actual habit of the user and the case is added to the case base that is used for location improvements. Using a moving window to keep track of possible cases, changing patterns over time will be appropriately added, removed and adjusted in the case base.

4.3. Evaluation Results

To demonstrate that case-based reasoning has an added value for the desk sharing use case and that the rule-based reasoning and case-based reasoning work very well together we performed some simulations. As test environment we simulated two floors of a typical office building (fig. 7Maps of the office environment that is used for the simulation. A simulated trajectory from the desk of a employee to a meeting room is shownfigure.7). The grid points constitute the possible locations of a user.

Table 1Simulation parameters for the desk sharing office use casetable.1 gives an overview of the simulation parameters we used for the experiments. For the location model, a location at a distance equal to the average error (μ_{error}) re-

Table 1. Simulation parameters for the desk sharing office use case.

Parameter	Description	Value
$P_l(\mu_{error})$	Location probability at distance equal to average error	$\frac{1}{4}$
μ_{error}	Average location error	$3.2m$
CBR threshold value	Threshold value to determine if a case is well-matching	0.6
τ	Used for the calculation of the time attribute match value	5 minutes
n_f	Number of fixed trajectories	8
n_r	Number of random trajectories	10
P_f	Probability of traveling a fixed traject	0.8
v_s	Simulated user speed	1.4 m/s
v	Motion model user speed	1.5 m/s

ceives a correctness probability that is four times smaller than the probability of the returned location by the location provider. As threshold value to determine if a case is well-matching, a value of 0.6 is used. Both values were determined by conducting a number of calibration experiments.

A number of working days at the office were simulated for a user. This user has a number of fixed trajectories (n_f) and a number of random trajectories (n_r). Every fixed trajectory has a certain probability (P_f) of being traveled on a certain day. The fixed trajectories represent the habits of the user (e.g. leave the building for lunch at 12.30 PM) while the random trajectories represent visits to colleagues, meetings, etc. Fig. 7Maps of the office environment that is used for the simulation. A simulated trajectory from the desk of a employee to a meeting room is shownfigure.7 shows an example trajectory for a user from his desk to a meeting room.

Instead of exact user speeds (v_s), a fixed assumed user speed (v) is used.

The random trajectories are added in pairs, from the last location of the user to a random destination and back. At the destination the user is standing still for a variable amount of time. These random trajectories will also influence the learning of the fixed trajectories. During the rest of the simulations, the user is standing still or sitting, mostly at his desk.

We simulated an error between $0m$ and $5m$ on the returned location by the location provider by randomly choosing a grid point within a range of $5m$ around the correct location. As a result the average error on the returned location (μ_{error}) is $3.2m$. This accuracy matches the earlier reported limit for localization inside a building using WiFi (Elnahrawy et al., 2004).

4.3.1. Localization Improvement

For this experiment we assume that the cases which represent the fixed trajectories of the user are already part of the case base. Results are averages over 20 simulated working days each consisting of 32 400 location updates.

During the simulations we calculate localization errors with and without the different improvements made by the reasoners (cfr. Algorithm 1Design Detailsalgocf.1). These calculations were split up taking the activity of the user (moving/standing still) or the structure of the building into account as for example the case-based reasoner will only improve locations for a moving user and the *improve_location_in_meeting*

Table 2. Localization improvement situations.

Localization Improvement	Remarks
Simulated location without any improvements	
Improved location (only rule-based motion model)	User is either 'moving' or 'standing still'
Improved location (rule-based motion model + inMeeting rule)	Only locations inside meeting rooms while the user is standing still are taken into account.
Improved location (rule-based motion model + case-based improvements)	Only locations for a user traveling a fixed traject are taken into account.

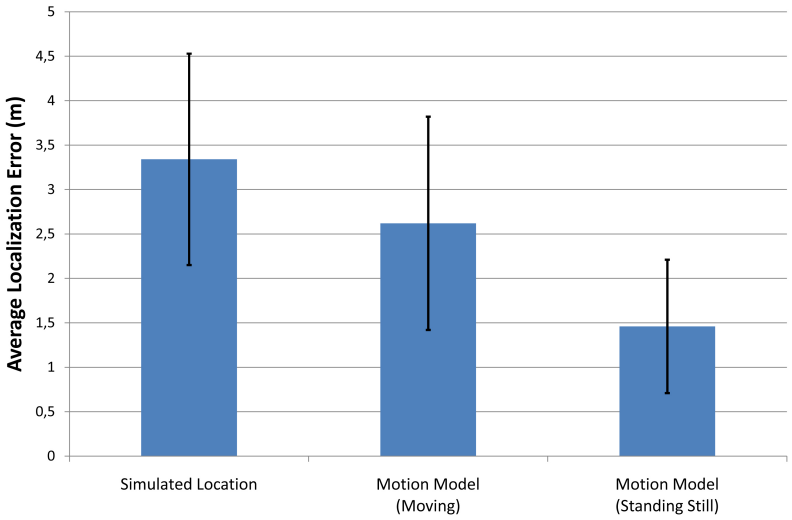


Fig. 8. Comparison of the localization error with and without rule-based motion model. Results are split up for a moving user and for a user which is standing still. The bars show the average values and the whiskers the standard deviation.

rule has only an impact for users inside meeting rooms. Table 2Localization improvement situationstable.2 gives an overview of the different situations for which localization errors were calculated.

We first focus on the impact of the rule-based motion model without using any other improvements. Figure 8Comparison of the localization error with and without rule-based motion model. Results are split up for a moving user and for a user which is standing still. The bars show the average values and the whiskers the standard deviationfigure.8 shows the resulting localization errors. Simulated errors are close to the earlier reported value of $3.2m$ (Elnahrawy et al., 2004).

Using the motion model we observe a large improvement in localization errors: 22% for a moving user and 56% for a user who is standing still. As our simple motion model assumes that a user is still at his previous location or somewhere

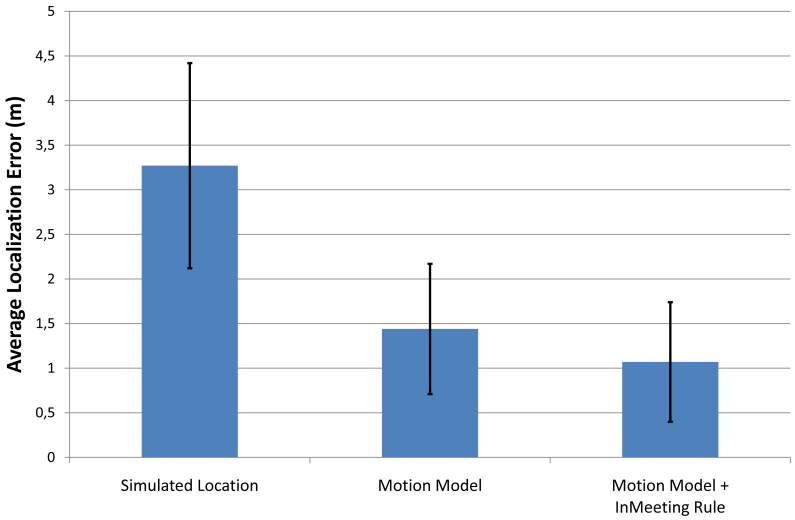


Fig. 9. Comparison of the localization error in meeting rooms for the different rule-based improvement techniques. The bars show the average values and the whiskers the standard deviation.

near to that location, obviously localization is better when the user is standing still.

Figure 9Comparison of the localization error in meeting rooms for the different rule-based improvement techniques. The bars show the average values and the whiskers the standard deviationfigure.9 shows the results when also the extra inMeeting-rule is used to improve localization in meeting rooms. The chart shows average localization errors for users that are standing still inside meeting rooms.

If we combine the motion model with the extra inMeeting rule we observe an extra improvement of 26% and an overall improvement of 67% in comparison to the simulated location.

The case-based reasoner only improves localization when a user is moving and follows a trajectory that is stored in the case base (in casu a trajectory the user follows often, at the same moment of the day). Therefore, to evaluate the effect of the case-based reasoner we now consider only localization errors during the traveling of such fixed trajectories. We implemented two methods to adjust the user location using the information from the matching case (Algorithm 1Design Detailsalgocf.1 lines 31-36). Method 1 considers the closest location on the path defined in the case. Method 2 does the same, but only considers locations on the path that are closer to the end location than the current location of the user. As indicated before, the implemented motion model favors locations close to the previous location of the user resulting in an improved location that often lags behind the actual location of the user. For this reason, we expect better results from Method 2. Figure 10Comparison of the localization error when the user is taking a fixed trajectory with with and without case-based reasoning in combination with the rule-based motion model. The bars show the average values and the whiskers the standard deviationfigure.10 shows the results: as expected the combination of the motion model with case-based reasoning - Method 1 only

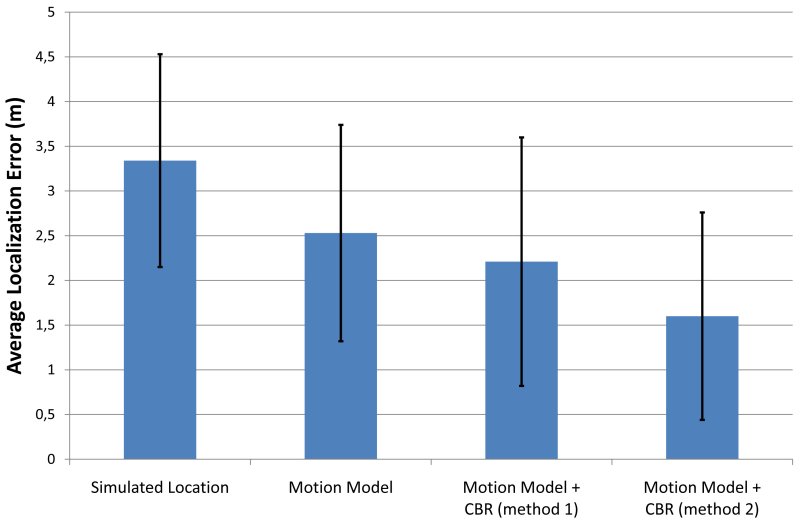


Fig. 10. Comparison of the localization error when the user is taking a fixed trajectory with with and without case-based reasoning in combination with the rule-based motion model. The bars show the average values and the whiskers the standard deviation.

Table 3. Case learning simulation parameters.

Parameter	Value
Number of simulations	5
Length simulation	50 working days
Maximum deviation start moment	2.5 minutes
Moving window size	10 working days
Minimum number of occurrences in window	5

shows a small improvement of about 13% in comparison to localization using only the motion model. Case-based reasoning - Method 2 shows a much larger improvement of 37% and an overall improvement of 52%.

4.3.2. Influence of Case Learning

In this experiment we investigate how well the fixed trajectories of a user can be learned. Therefore, we perform a number of long running simulations, giving the system the time to learn the habits of the user and to evaluate automatically whether localization improves in a similar way as in the previous test.

Table 3Case learning simulation parameterstable.3 shows the simulation parameters we used for this test. Due to the simulated error on the returned locations of a user, some variation on the start and end locations of the user's trajectories is already present in the simulations. We also varied the start moment of a trajectory within a time interval of 5 minutes. When a possible case occurs at least 5 times in a moving window, the average value of these grouped possible cases is considered as a learned case and added to the case base.

When evaluating the case base at the end of the simulations, we observe that always 7 or 8 of the 8 simulated fixed trajectories are present which corresponds with a success rate of at least 87.5 % for this use case. Now and then also one

Table 4. Average and maximum errors for the different attributes of the learned cases.

Case attribute	Average error (m)	Standard deviation (m)	Maximum error (m)
Start point	1.64	0.17	6.26
End point	2.17	0.26	3.5

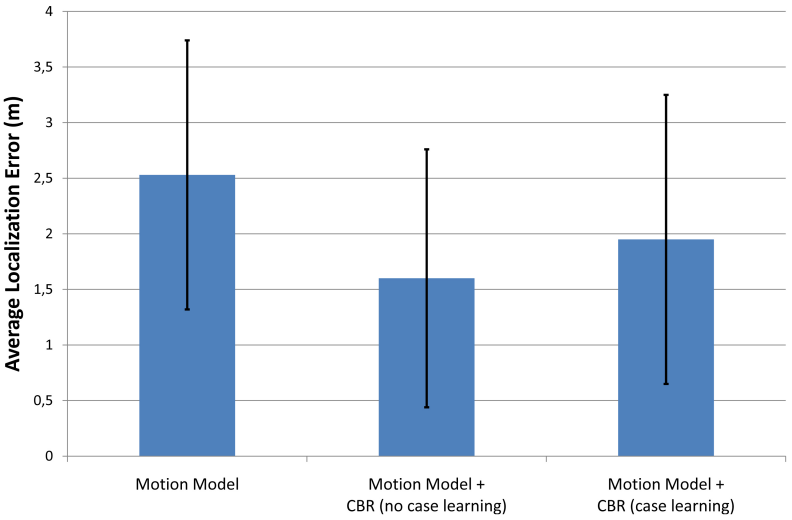


Fig. 11. Comparison of the localization error when the user is taking a fixed trajectory with case learning and without case learning. The bars show the average values and the whiskers the standard deviation.

extra case is present due to the random trajectories of the user. These results are very good as possible cases representing one fixed trajectory are sometimes split up over 2 possible case groups or they are grouped with possible cases representing random trajectories.

Table 4Average and maximum errors for the different attributes of the learned casestable.4 shows the average and maximum errors on start and end points of the cases that are present in the database at the end of the 5 simulations. We can note that the average error on the end point is larger than for the start point. The reason is that before a user starts traveling along a trajectory he is standing still, while at the end of the trajectory the user is moving. As shown in the previous section the used rule-based localization improvement works better when standing still than when moving. The maximum error for the start point is quite large, but detailed analysis of this error shows that the possible case group contained a possible case from a random trajectory with well-matching attributes for 2 of the 3 case attributes. The third case attribute (start point respectively end point) didn't match very well and had a negative impact on the average value of the possible case group.

Figure 11Comparison of the localization error when the user is taking a fixed trajectory with case learning and without case learning. The bars show the average values and the whiskers the standard deviationfigure.11 shows the localization errors for the fixed trajectories. In section 4.3.1Localization Improvementsubsubsection.4.3.1 we obtained an average error of $2.53m$ without case-based reasoning and $1.60m$

Fig. 12. Use case overview: two users chat with each other using a web based enriched chat client

with case-based reasoning and all cases present in the case base. When the cases still have to be learned during the tests we see now an average error of $1.95m$. This error is of course higher than with all cases present from the start due to the time that is needed to learn the user's habits. During our simulations of 50 working days, about 66% of locations on the fixed trajectories are improved using case-based reasoning. In the previous experiments about 95% of the locations were improved. The learned cases are also not perfect matches of the actual trajectories of the user as there are errors on the start and end points (and corresponding shortest paths) as shown in table 4Average and maximum errors for the different attributes of the learned casestable.4. This gives an additional error. However, localization is 23% better than when no case-based reasoning is used and 42% better overall. In conclusion, the system succeeds in automatically learning the cases and using these cases to improve localization.

5. Use Case: Enhanced Instant Messaging Communication Service

5.1. Motivation

In this use case users of an instant messaging client are provided with content that is an added value to their conversation, i.e. pictures about the topic they are discussing that match at the same time with their personal interests. These pictures enrich the conversation and will possibly influence it.

Figure 12Use case overview: two users chat with each other using a web based enriched chat clientfigure.12 gives an overview of the presented use case. Two users chat with each other using a web based chat client. Each user has an associated user profile model containing a number of keywords with associated weight values that indicate the importance of a keyword for the user. The contents of the conversation are redirected to a text analyzer, which is able to recognize a certain set of keywords, based on the topic of the conversation and the weight values of the keywords in the models.

When a keyword is recognized, the keyword model of the user is updated. At the same time pictures are looked up in a content store (e.g. Flickr) based on the recognized keyword and related keywords in the keyword model.

The resulting pictures are sent to the instant messaging clients and added in a rotating carousel of pictures. A user can click on any picture in the rotating list to see a larger version. In that case the keyword model for that user is updated with the feedback.

Figure 13Screenshot of the user interface - trained modelfigure.13 provides

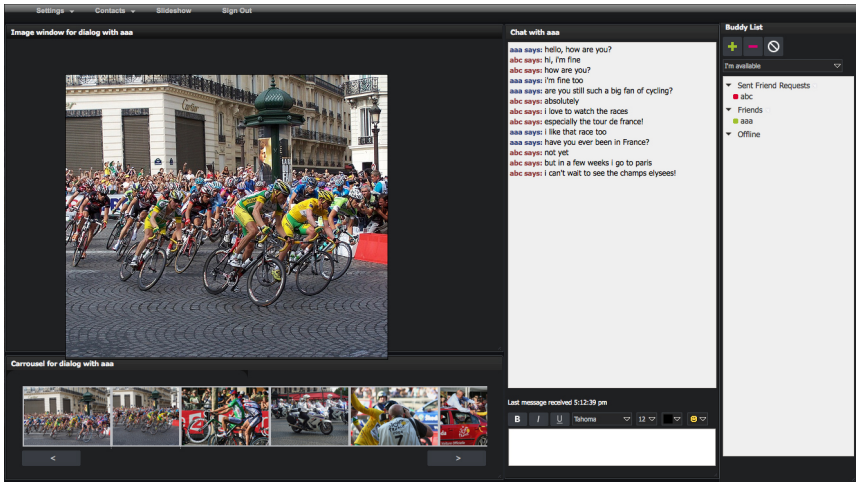


Fig. 13. Screenshot of the user interface - trained model

a screenshot of the user interface of the developed use case showing the online buddies on the right, the chat conversation in the middle, a carousel of 10 pictures at the bottom left and the picture the user most recently clicked on the left. This figure also illustrates the effect of a trained keyword model on the provided pictures. The users start by talking about cycling and then about the Champs Elysées in Paris. As a result pictures about the traditional last stage of the Tour de France on the Champs Elysées are shown. With an untrained model the typical travel guide pictures of the Champs Elysées are shown with the Arc de Triomphe, traffic by night, etc.

5.2. Design Details

5.2.1. Keyword Models

To represent the user interests we developed two keyword models (Strobbe et al., 2010): a simple keyword tree and a more expressive keyword ontology. In both approaches keywords are modeled using a tree structure with keywords representing main categories on top and subcategories and specific interests lower in the tree. Each node in the tree is characterized by a (normalized) weight value. This weight value represents the user specific importance of the keyword. A formal representation is shown in figure 14Formal representation of the keyword models - the *comesNext* and *usedTogether* relationships are only part of the keyword ontologyfigure.14. Each node $n_{i,j}$ has an index i, a parent node j and a weight value $w_{i,j}$. The sum of all keywords on any level in the tree is normalized to 1:

$$\sum_{i} w_{i,j} = 1 \tag{8}$$

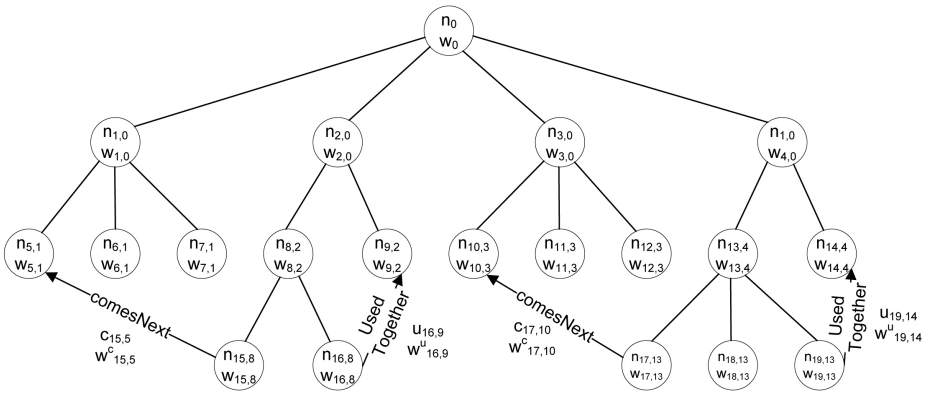


Fig. 14. Formal representation of the keyword models - the *comesNext* and *usedTogether* relationships are only part of the keyword ontology

These weight values are adapted when input or feedback is received from the user and are used to select appropriate keywords to search for content.

As a second model, we designed an ontology based on the keyword tree model, modeling a number of additional relationships between the keywords to make the model more expressive and thereby more useful for applications.

The *comesNext* relationship indicates that when keyword i is used, often keyword j (belonging to another branch) follows within a short time frame. This relationship could be used to predict which subset of the keyword tree will be used in the near future by an application. This might be important when for example only a limited set of the keywords can be stored in memory, due to memory or real-time constraints, or to prefetch relevant content. A formal representation is shown on figure 14Formal representation of the keyword models - the *comesNext* and *usedTogether* relationships are only part of the keyword ontologyfigure.14. Each instance of this relationship $c_{i,j}$ connects node $n_{i,x}$ with node $n_{j,y}$ and has an associated weight value $w^c_{i,j}$ between 0 and 1 indicating the strength of the relationship.

The *usedTogether* relationship indicates that certain keywords (belonging to the same branch) are often used together. This relationship might be useful when selecting additional keywords (besides the recognized one) for a personalized content search (query expansion). Using a keyword tree only high valued keywords that are related to the recognized keyword via a parent, child or sibling relationship are used for this purpose. This relationship allows keywords that are often used together with the input keyword, but are not closely related, to be used in addition to the related keywords. In general, each relationship $u_{i,j}$ connects node $n_{i,x}$ with node $n_{j,y}$ and has also an associated weight value $w^u_{i,j}$ between 0 and 1 indicating the strength of the relationship.

5.2.2. Designed Ontology

Figure 15Designed ontology for the enhanced instant messaging use casefigure.15 shows the ontology used for this use case, which is an extended version of the ontology we presented in (Strobbe et al., 2010). Additional personal and context information about the user is modeled: his home and work address, his current

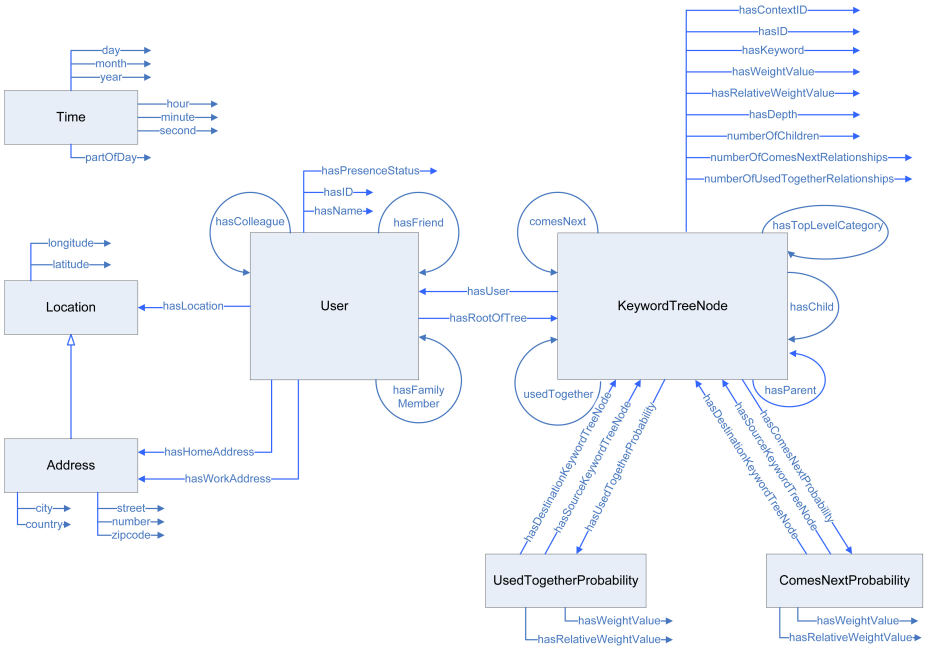


Fig. 15. Designed ontology for the enhanced instant messaging use case.

location, a presence status based on the location of the user and the time of the day. This extra information is used by the reasoners to improve the retrieval of matching content (see section 5.2.5Case-Based Reasoningsubsubsection.5.2.5). Note that when the keyword tree is used for modeling the user interests, only the left part of the ontology is used.

5.2.3. Implemented Algorithms

We designed a number of algorithms for adapting the weight values in the interest models based on user input and feedback, for learning the user interests, for using the profile information for retrieving content and assisting the user input analysis process. Figure 16Overview of the interactions between the algorithmsfigure.16 shows a general overview of the interactions between the implemented algorithms.

When a user communicates with another user, the words he uses are an indication of his interests. This input is analyzed by a *Recognizer*. Based on the available set of keywords in this component, words will be recognized and their associated weight values will be updated.

In the case of the ontology based keyword model, also the *ComesNextProbabilities* and *UsedTogetherProbabilities* concepts in the model are adapted by consulting the recently recognized keywords within a certain time frame.

The *Keyword Selection for Content Search Algorithm* constructs a set of relevant search terms for the communication at hand and retrieves matching content from the content repository. The selection of keywords is based on the recognized keyword, and the ones related to it. For the tree based keyword model these re-

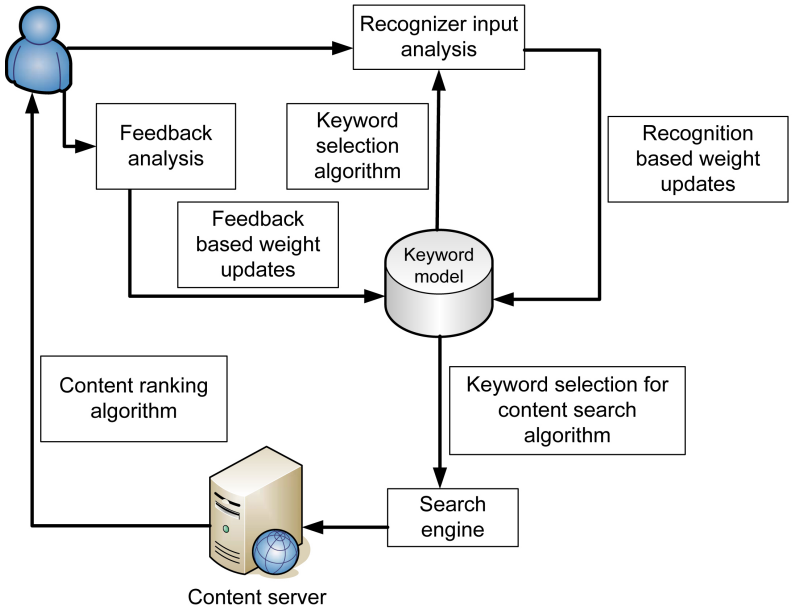


Fig. 16. Overview of the interactions between the algorithms

lated keywords consist of the parent keyword and sibling or child keywords. In the case of the ontology based model also keywords that are often used together (via the *usedTogether* relationship) are added.

The retrieved content items are ranked by the *Content Ranking Algorithm* based on the goodness of the match between the keywords used for the search and the attached tags.

The ranked results (URLs) are then sent to the client of the user. When the user clicks on a result feedback is provided to the *Feedback Analysis Algorithm* which updates the weight values of the attached tags that are part of the keyword model accordingly.

Finally, the *Keyword Selection Algorithm* will identify a subset of keywords from the model to be used in the recognizer in case of real-time or memory constraints. It would possibly require too much memory or processing power to recognize keywords for a large vocabulary, for example when speech recognition is used or when a server needs to handle lots of users. The subset contains mostly related keyword to recent recognized keywords. Related keywords are children, parent, sibling keywords but in case of the ontology model also keywords connected via *usedTogether* and *comesNext* relationships as these relationships indicate that the related keywords have a high probability of being used in the near future.

More details about the designed algorithms and the used formulas can be found in (Strobbe et al., 2010).

5.2.4. Rule-Based Reasoning

The ontology based model allows to infer new information using the rule reasoner of CASP. The *UsedTogetherProbability* and *ComesNextProbability* concepts in the ontology model the fact that *usedTogether* and *comesNext* relationships might exist between two keywords. Only when these concepts have a relative high weight value there is a real relationship present in the model which is useful for applications. The deduction of the actual relationships is done by rules.

As an illustration the following rule will infer an actual *comesNext* relationship when the ontology contains a *ComesNextProbability* concept with a relative weight value above average (larger than 1).

```
[comesNextRelation:
  (?s comesNext ?d)
  <-
  (?x rdf:type ComesNextProbability)
  (?x hasRelativeWeightValue ?y)
  greaterThan(?y '1.0')
  (?x hasSourceKeywordTreeNode ?s)
  (?x hasDestinationKeywordTreeNode ?d)]
```

Note that a *relative weight value* is defined as the actual weight of a keyword times the number of children on the level of the keyword:

$$w^r_{i,j} = w_{i,j} \times \sum_i n_{i,j} \tag{9}$$

This allows to compare weight values of nodes in the tree in a correct way: a node with value 0.25 having 8 siblings, has a higher importance than a node with weight 0.3 having only one sibling (which means its sibling has a weight of 0.7). Obviously, the average relative weight value of a node is always 1. Using rules allows to easily change the thresholds that define the minimal strenghts for the relationships.

We also use rules to abstract the location of the user to a presence status saying that the user is for example 'atWork' or 'atHome'. The rules look at the locations of the user and match these with the coordinates of known places for the user like his home or work address.

5.2.5. Case-Based Reasoning

Users often chat with each other in different contexts. For example at work with colleagues about work related topics and in the evening or during weekends with family and friends about personal topics. As people talk about other things in those different contexts, their interests are also dependent on the context. Instead of using one interest model per user to retrieve matching content, better results can be achieved by using an interest model per context. These contexts are different for every user and might change over time. Therefore, the system should be able to recognize and learn different chat contexts.

In order to accomplish this, we use the case-based reasoner of CASP using the algorithm in pseudocode below (Algorithm 2Case-Based Reasoningalgocf.2). The cases in the case base model the different contexts of the user taking into account the user's presence status, the time of the day and the kind of relationship

Table 5. Possible values for the different context attributes (presence status, time of the day, chat partner relationship).

Attribute	Values
Presence Status	atWork, atHome, unknown
Time of the day	morning, afternoon, evening, night
Relationship	colleague, family, friend

between the user and his chat partner. When a user starts a new conversation, the case base is queried for a matching case (line 6). If found, the corresponding keyword model is used to retrieve matching content (line 8).

```
1  Upon start of new conversation for user u
2  begin
3  |  p ← KB // Retrieve user's presence status
4  |  t ← KB // Retrieve time of the day
5  |  r ← KB // Retrieve relationship with chat partner
6  |  matchingCase = searchMatchingCaseInCaseBase(u, p, t, r)
7  |  if matchingCase != null then
8  |  |  u.setCurrentContextID(matchingCase.getContextID)
9  |  end
10 |  else
11 |  |  u.setCurrentContextID("default")
12 |  end
13 |  caseManager.addNewCaseInfo(u, p, t, r)
14 end
```

Algorithm 2: Learning and using context dependent user interests via case-based reasoning.

To determine whether a case is well-matching, the user, the user's presence status, the time of the day and the chat partner relationship attributes are compared with the chat context at hand. For these 4 attributes, a match value is calculated and the average value of these match values is compared with a threshold. The calculation of the match values for the attributes is straightforward for this use case: 1 for an exact match and 0 otherwise. Table 5Possible values for the different context attributes (presence status, time of the day, chat partner relationship)table.5 shows the possible values for the status, time of the day and relationship attributes we used for the experiments. The number of possible values is discrete and limited, so we imposed that a case from the case base needs to match perfectly with the situation at hand.

At the same a new possible case is added to the case manager (line 13). When a user often chats in a similar context, that context is added to the case base and a new keyword model is associated with that context. Again we use a moving window to recognize changing patterns over time.

If no well-matching case is found in the case base, a default keyword model is used (line 11).

Table 6. User input and feedback simulation parameters.

Parameter	Value
Content store size	5,000 items
Number of tags per content item	10
Number of keywords in model	500
Number of contexts	2
Number of runs per simulation	2
Number of generated keywords per context	200
Number of retrieved content items per search	5
Number of simulations	20

5.3. Evaluation Results

In order to evaluate the added value of case-based reasoning for this use case, a number of simulations were performed. Arbitrary keyword models were generated and populated with code words. The size of the model was specified and fixed, depending on the experiment. The code words are unique string identifiers generated as all possible combinations of the letters of the alphabet (e.g. 'aa', 'ab', ...).

The simulated users alternately chat in a specified number of different contexts. In each context they have different interests, randomly chosen from the keyword model. Similarly, in each context, a number of keywords are selected which are often used together by the simulated users and for each topic a number of topics that typically follow next in the conversation.

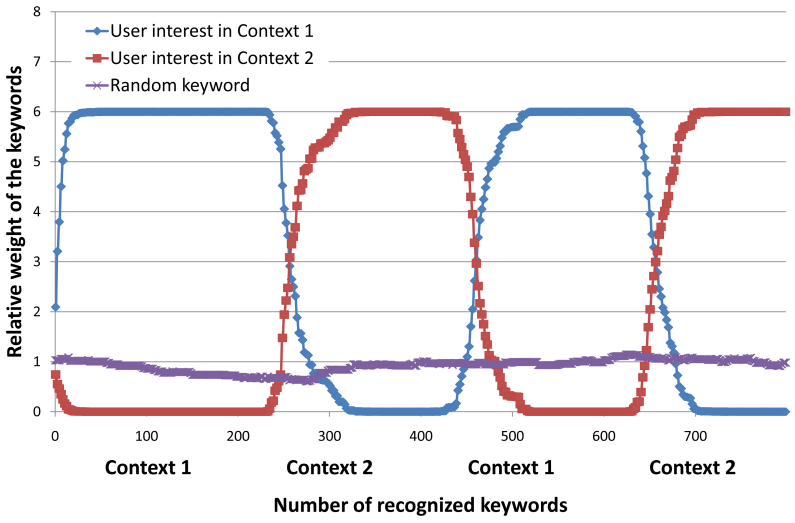
5.3.1. User Input and Feedback Algorithm

The algorithms take user input and feedback into account to learn the specific interests of a user. In this experiment the impact of this user input and feedback is verified with and without context dependent keywords models. As a user might have different interests in different contexts we expect to see that his interests are retrained over and over again when only one keyword model for all possible contexts is used (i.e. without case-based reasoning). With separate keyword models per context, the weight values will keep their correct value after a context switch.

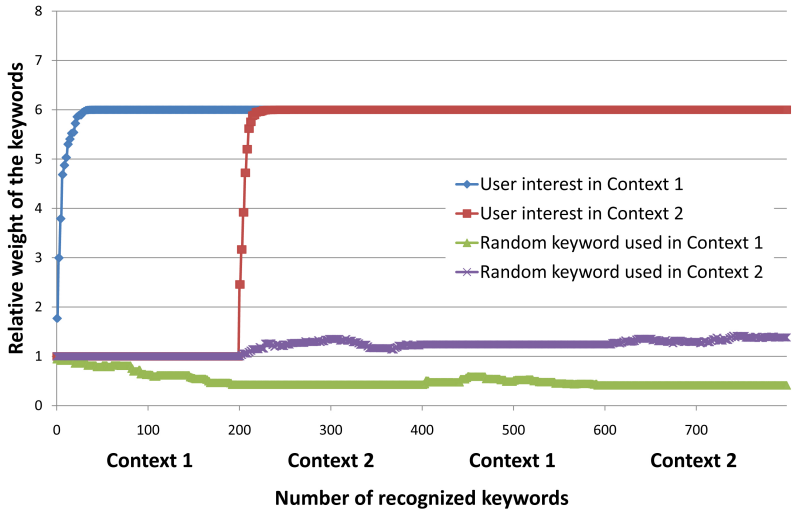
For this experiment we assume that the different contexts are perfectly recognized by the case-based reasoner. Table 6User input and feedback simulation parameterstable.6 shows the used simulation parameters. One simulation consists of two runs of alternately chatting in two different contexts.

Two sibling keywords were tracked, with one keyword being an user interest in *Context 1* and the other being an interest in *Context 2*. For these keywords feedback is given by the user when content containing this keyword as tag, is presented to him. For this purpose we simulated a content repository with a number of content items, each containing 10 random tags. Besides the user interests also a randomly chosen keyword is tracked that is no user interest nor sibling of a user interest.

Figures 17(a)Subfigure 17(a)subfigure.17.1 and 17(b)Subfigure 17(b)subfigure.17.2 show the results. The weight values are relative weight values as explained in section 5.2.4Rule-Based Reasoningsubsubsection.5.2.4. The keywords that represent actual interests of the user get very high weight values, but with only one keyword model for both contexts, the weight values of the interests fluctuate



(a) One keyword model



(b) Keyword model per context

Fig. 17. Comparison of keyword weights with and without context dependent keyword models.

between these high values and very low values. They need to be retrained at the start of every new conversation. With a keyword model per context the weight values keep their high value after being trained once.

The random keyword stays around the average weight value of 1 as it just undergoes its recognitions, in a similar way as all other keywords.

Table 7. Keyword selection for content search simulation parameters.

Parameter	Value
Content store size	20,000 items
Number of tags per content item	10
Number of user interests per category	5
Number of search keywords	4
Number of *usedTogether* keywords	1 to 5
Number of keywords in model	125
Number of training runs per simulation	5
Number of test runs per simulation	10
Number of generated keywords per context	50
Number of retrieved content items per search	25
Number of simulations	3

5.3.2. Keyword Selection For Content Search Algorithm

For this experiment we also assume that the different contexts are perfectly recognized by the case-based reasoner. Table 7Keyword selection for content search simulation parameterstable.7 shows the used simulation parameters. We simulated a content repository and for each context we randomly selected 5 user interests per top-level category. If content is retrieved from the content store with a tag matching one of these interests, user feedback is given. Three extra keywords (besides the recognized keyword) are used for the content search. When testing the keyword ontology a set of *usedTogether* keywords was predefined for every keyword and for every context. During the simulations we assumed a 50% probability that a keyword from this set was generated as next keyword. The presence of these *usedTogether* relationships in the models is important as the keyword selection for content search algorithm checks these relationships to determine the set of search keywords.

As *comesNext* relationships are not relevant for this algorithm we evaluated for only one top-level category consisting of 125 keywords. One simulation consists of 5 training runs and 10 test runs of alternately chatting in a number of different contexts. The training runs ensure that there are already some *usedTogether* relationships present in the models when the test runs start. In each context 50 keywords were randomly generated and for each keyword 25 content items were retrieved from the content store. For each content item it was checked if one or more of the tags match with the defined user interests.

Users will typically use the keywords they are really interested in, often during a conversation. This was simulated by defining an interest probability that indicates the chance a simulated keyword is a predefined user interest. Tests for different interest probabilities ranging from 0% till 100% were performed.

Figure 18Comparison of the number of content items that match user interests for a user chatting in 10 different contexts. As user profile a keyword ontology model is usedfigure.18 shows the percentage of content items that match at least one or several user interests for a user chatting in 10 different contexts. As user profile the ontology keyword model is used and the figure shows the results with one model for all different contexts (no case-based reasoning used) and with separate models per context and this for different values of the interest probability.

Results are clearly better when several keyword models are used for the user profile, especially the number of content items matching more than one interest is

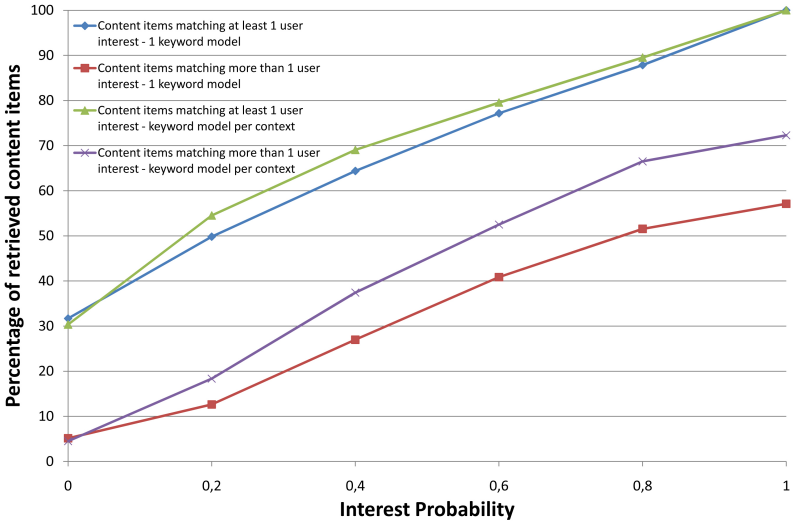


Fig. 18. Comparison of the number of content items that match user interests for a user chatting in 10 different contexts. As user profile a keyword ontology model is used.

Table 8. Standard deviations for figure 18Comparison of the number of content items that match user interests for a user chatting in 10 different contexts. As user profile a keyword ontology model is usedfigure.18

Interest Probability	0.0	0.2	0.4	0.6	0.8	1.0
Content items matching at least 1 user interest - 1 keyword model	0.63	1.09	0.68	0.48	0.81	0.0
Content items matching more than 1 user interest - 1 keyword model	0.17	0.27	0.33	3.44	4.84	0.81
Content items matching at least 1 user interest - keyword model per context	0.74	1.05	0.49	1.45	0.42	0.0
Content items matching more than 1 user interest - keyword model per context	0.14	0.23	0.60	2.26	2.26	1.12

higher. For higher interest probabilities lots of keywords will have a *usedTogether* relationship with these often generated interests after a while. When only one keyword model is used for all chat contexts sometimes user interests from other chat contexts will be used for the content search as the weights in the model need to be retrained over and over again. Especially at the beginning of a conversation in a new context, the interests of the previous context will still have hight weight values and the *usedTogether* relationships with these interests will still be intact. This results in less content items matching the current interests of the user.

The tree based keyword model has only a limited benefit of the case-based reasoning as only sibling and child keywords of the recognized keyword are added to the set of keywords that are used for the content search. Without the presence of *usedTogether* relationships, the probability that an interest of the previous context is used for the search is small. Figure 19Comparison of the number of content items that match user interests for a user chatting in 10 different contexts. As user profile a keyword tree model is usedfigure.19 illustrates this. It shows the results for a user chatting in 10 different contexts with a user profile consisting of a single keyword tree model and consisting of a model per context.

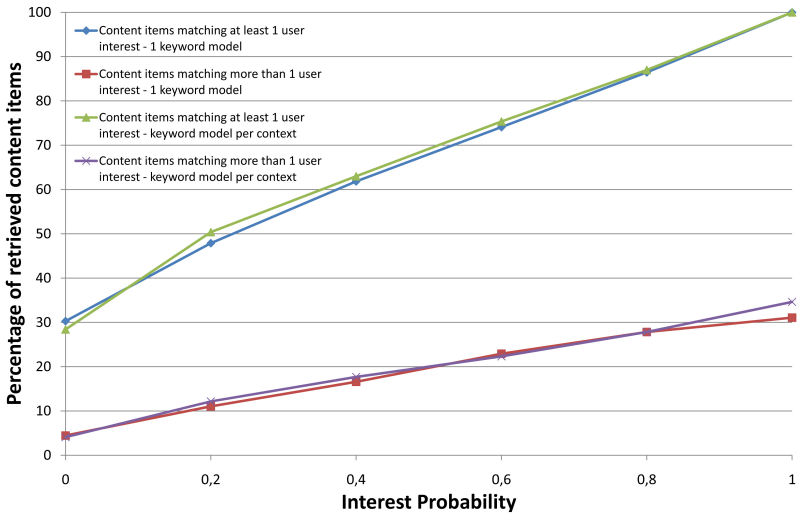


Fig. 19. Comparison of the number of content items that match user interests for a user chatting in 10 different contexts. As user profile a keyword tree model is used.

Table 9. Standard deviations for figure 19Comparison of the number of content items that match user interests for a user chatting in 10 different contexts. As user profile a keyword tree model is usedfigure.19

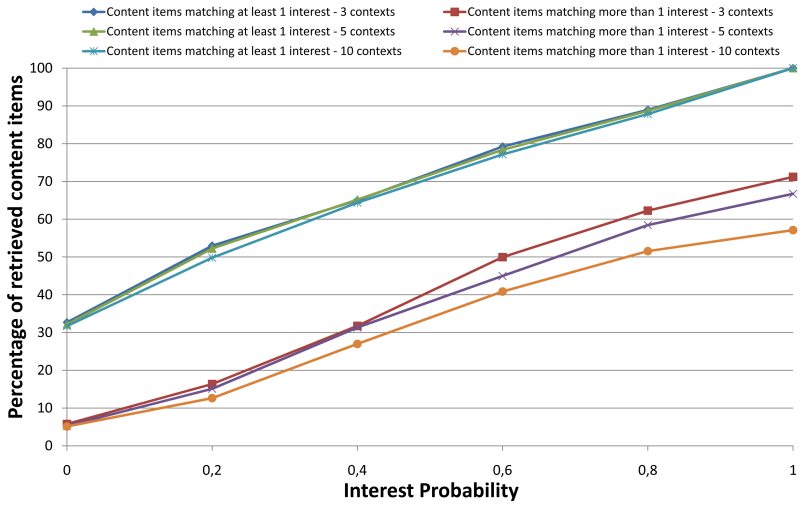
Interest Probability	0.0	0.2	0.4	0.6	0.8	1.0
Content items matching at least 1 user interest - 1 keyword model	0.59	1.21	1.12	0.68	0.67	0.0
Content items matching more than 1 user interest - 1 keyword model	0.11	1.07	0.19	2.64	0.90	5.41
Content items matching at least 1 user interest - keyword model per context	0.89	1.04	0.59	1.11	0.23	0.0
Content items matching more than 1 user interest - keyword model per context	0.27	0.34	1.06	1.44	1.81	2.49

Figures 20(a)Subfigure 20(a)subfigure.20.1 and 20(b)Subfigure 20(b)subfigure.20.2 show the effect of the number of different contexts for the ontology keyword model. Simulations were now performed for 3, 5 and 10 different contexts. With a user profile model per context, the number of content items that match with the user's interests is always more or less the same. If there is only one keyword model, the number of content items matching several interests decreases with increasing number of contexts. With a lot of different contexts and corresponding user interests, there is a relatively high chance that an interest from a previous context still has a high weight value and is wrongly used for the content search.

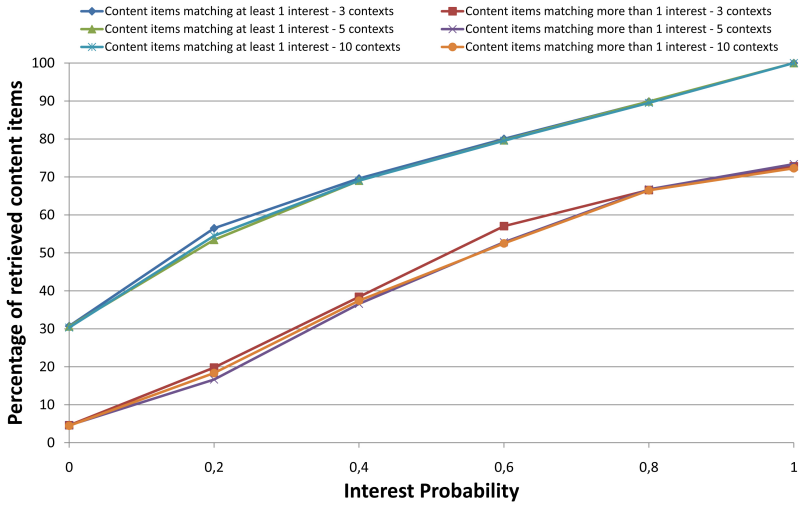
5.3.3. Keyword Selection Algorithm

In this experiment we evaluate the Keyword Selection Algorithm which selects a relevant subset of keywords from the keyword model, with and without context dependent keywords models.

As the goal of the use case is to enhance communication sessions with appropriate content, it is particularly important that the shown content closely follows



(a) One keyword model



(b) Keyword model per context

Fig. 20. Comparison of the number of content items that match user interests for a user chatting in different numbers of contexts. As user profile a keyword ontology model is used.

the topic of the conversation for a good user experience. When a user's keyword model contains the most popular keywords for the defined categories and enough resources are available to analyze the conversation for all keywords in the model, keywords will be often recognized and the content will nicely track the conversation. In situations where only a subset of the keyword model can be used (here called *current keyword list*), e.g. because a server handling a lot of users, can only recognize a limited number of words in real-time due to memory or CPU constraints, the selection of this subset has to be adaptive to topic changes.

Table 10. Keyword Selection Algorithm parameters.

Parameter	Value
Number of different contexts	3
Number of generated keywords per context	250
Number of generated keywords per topic	50
Number of *usedTogether* keywords	1 to 5
Number of *comesNext* keywords	1
Number of training runs per simulation	20
Number of test runs per simulation	50
Current keyword list size	100
Number of user interests per category	5
Interest probability	0.5
Number of simulations	5

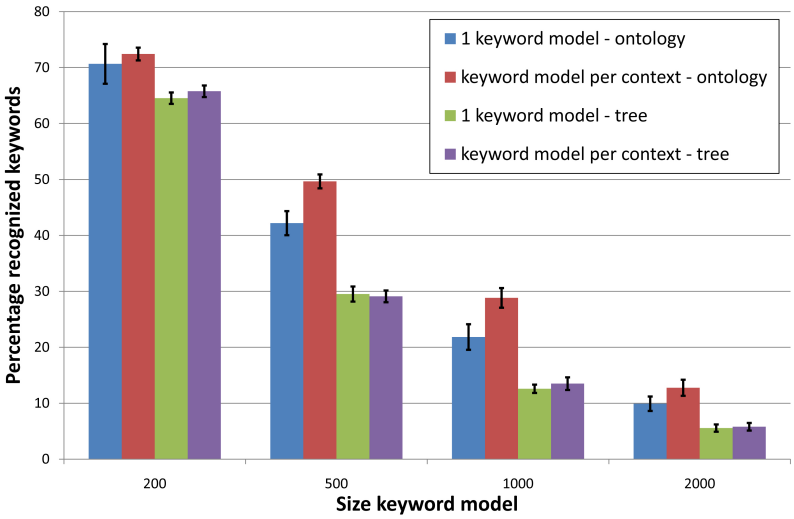


Fig. 21. Comparison of the performance of the Keyword Selection Algorithm with and without context dependent keyword models. The bars show the average values and the whiskers the standard deviation.

For this experiment we also assume that the different contexts are perfectly recognized by the case-based reasoner. Table 10Keyword Selection Algorithm parameterstable.10 shows the used simulation parameters. Conversations were simulated by generating random keywords from the keyword model. Consecutive keywords come from the same main category and every 50 keywords a category switch occurs to simulate a change in the conversation topic. When testing the keyword ontology a set of *usedTogether* keywords and *comesNext* keywords was predefined for every keyword and for every context. One simulation consists of a number of training and test runs of alternately chatting in three different contexts. The training runs ensure that there are already some *comesNext* and *usedTogether* relationships present in the models when the test runs start. To assure the presence of enough *usedTogether* relationships, just as in the previous experiment, we used a predefined set of user interests per category and per context, combined with an interest probability which indicates the probability that a simulated keyword is such a predefined interest.

Table 11. Case learning simulation parameters.

Parameter	Value
Number of simulations	5
Length simulation	50 days
Number of predefined contexts	5
Number of random contexts	2
Probability of using predefined chat context	0.8
Moving window size	10 working days
Minimum number of occurrences in window	5

Figure 21Comparison of the performance of the Keyword Selection Algorithm with and without context dependent keyword models. The bars show the average values and the whiskers the standard deviationfigure.21 shows the results for different sized keyword models for a user having a current keyword list of size 100. The figure shows the results with one model for all different contexts (no case-based reasoning used) and with separate models per context, for both tree based and ontology based keyword models.

When observing the results for the ontology keyword model we note that the results are better with a keyword model per context. With only one keyword model sometimes *usedTogether* and *comesNext* relationships that belong to another context than the current context will be used to fill the current keyword list of the user. This is again a consequence of the fact that the user profile is retrained over and over again and results in less recognized keywords.

For the the tree based keyword model, the differences between the results are small due to the lack of *usedTogether* and *comesNext* relationships.

As already shown in (Strobbe et al., 2010), this figure also illustrates the added value of the extra *usedTogether* and *comesNext* relationships in the ontology keyword model. The results are clearly better than for the tree based model.

5.3.4. Case Learning

In this experiment we study how good the chat contexts of a user are learned. Therefore, we perform a number of long during simulations, giving the system the time to learn the typical chat habits of the user and to test if the retrieval of matching content improves in a similar way as in the previous experiments.

Table 11Case learning simulation parameterstable.11 shows the simulation parameters we used for this experiment. We simulated 50 days consisting of 5 predefined chat contexts (with a probability of 0.8 of actually being used that day) and 2 random contexts. Again when a possible case occurs at least 5 times in a moving window, the average value of these grouped possible cases is considered as a learned case and added to the case base. Possible cases are only grouped when they perfectly match with each other. With these settings for learning chat contexts, we expect to see few random cases in the case base. It's important that only real habit chat contexts are learned, so that the profile of a user doesn't contain too much distinct keyword models. This would result in a bad performance of the system and it would take a long time before the models are sufficiently trained.

If we observe the case base at the end of the simulations we note that the 5 predefined chat contexts are always present. Only in one experiment one extra random chat context was present in the case base.

Table 12. Comparison of the number of content items that match user interests for a user chatting in 3 different contexts and an interest probability of 0.6. Chat contexts are learned during the simulations.

	Content items matching at least 1 user interest		Content items matching more than 1 user interest	
	Average (%)	STDV (%)	Average (%)	STDV (%)
Case learning (keyword model per context)	77.75	0.75	35.95	1.05
No Case learning (1 keyword model)	75.86	0.39	32.81	1.19

During the simulations, we repeated our first experiment, evaluating the number of content items matching one or more user interests. We used the same parameters as in table 7Keyword selection for content search simulation parameterstable.7 and a value of 0.6 for the interest probability. Results are shown in table 12Comparison of the number of content items that match user interests for a user chatting in 3 different contexts and an interest probability of 0.6. Chat contexts are learned during the simulationstable.12 and are again better with a keyword model per context. Due to the random contexts the results are somewhat lower than in the previous experiment. For these random contexts always the same default keyword model is used which is retrained over and over again. As a consequence, we can state that the system succeeds in automatically learning the chat contexts of a user and using these to improve content retrieval.

6. Conclusions and Future Work

In this paper we discussed the design of a context platform incorporating rule-based as well as case-based reasoners. Applicability of this hybrid approach is illustrated in two application domains. In the desk sharing office environment use case the reasoners are used to improve the localization of employees. Individual trajects of users are learned by the case-based reasoner, while general knowledge such as 'a user with status *inMeeting* will normally be in a meeting room' is represented by rules. In the enhanced instant messaging use case, case-based reasoning is used to detect the context of the user who is chatting and the rule-based reasoner derives the comesNext and usedTogether relationships between keywords in the interest models of the users. The use cases show that case-based reasoning succeeds in learning trends and habits.

The rules and algorithms used to improve the system performance in the presented use cases are primarily conceived to investigate the added value of adding a case-based reasoner to a rule-based approach. For example, the rule-based motion model we implemented for the desk sharing office environment use case could be extended to include speed and direction measurements of the user.

To further increase the performance of the systems, user experiments could be conducted to assess the quality of the response as perceived by the users.

As future work, we plan to evaluate the performance and applicability of fuzzy reasoning.

Acknowledgements. Matthias Strobbe is a research assistant of the Fund for Scientific Research - Flanders (F.W.O.-Vlaanderen).

References

Aamodt, A. and Plaza, E. (1994), 'Case-based reasoning: foundational issues, methodological variations, and system approaches', *AI Commun.* **7**(1), 39–59.

Apt, K. (2003), *Principles of Constraint Programming*, Cambridge University Press, New York, NY, USA.

Baralis, E., Cagliero, L., Cerquitelli, T., Garza, P. and Marchetti, M. (2010), 'Cas-mine: providing personalized services in context-aware applications by means of generalized rules', *Knowledge and Information Systems* pp. 1–28.

Bouamrane, M.-M., Rector, A. and Hurrell, M. (2010), 'Using owl ontologies for adaptive patient information modelling and preoperative clinical decision support', *Knowledge and Information Systems* pp. 1–14.

Chong, S., Gaber, M., Krishnaswamy, S. and Loke, S. (2011), 'Energy conservation in wireless sensor networks: a rule-based approach', *Knowledge and Information Systems* pp. 1–36.

Clark & Parsia (2010), 'Pellet: Owl 2 reasoner for java', online.

De Paz, J., Bajo, J., Gonzlez, A., Rodrguez, S. and Corchado, J. (2010), 'Combining case-based reasoning systems and support vector regression to evaluate the atmosphereocean interaction', *Knowledge and Information Systems* pp. 1–23.

Dey, A. K. (2000), Providing architectural support for building context-aware applications, PhD thesis, Georgia Institute of Technology.

Ejigu, D., Scuturici, M. and Brunie, L. (2008), 'Hybrid approach to collaborative context-aware service platform for pervasive computing', *JCP* **3**(1), 40–50.

Elnahrawy, E., Li, X. and Martin, R. P. (2004), The limits of localization using signal strength: a comparative study, pp. 406–414.

Fact++ (2009), 'Fact++ owl-dl reasoner.', http://owl.man.ac.uk/factplusplus/.

Fox, D., Hightower, J., Liao, L., Schulz, D. and Borriello, G. (2003), 'Bayesian filtering for location estimation', *IEEE Pervasive Computing* **02**(3), 24–33.

GAIA - Group for Artificial Intelligence Applications (2010), 'jCOLIBRI Case-Based Reasoning Framework', http://gaia.fdi.ucm.es/projects/jcolibri/.

Garca, J. A. R. (2008), jCOLIBRI: A multi-level platform for building and generating CBR systems., PhD thesis.

Gartner (2009), 'Gartner says context-aware computing will be a $12 billion market by 2012.', http://www.gartner.com/it/page.jsp?id=1229413.

Glasgow, J., Narayanan, N. H. and Chandrasekaran, B., eds (1995), *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, AAAI Press, Menlo Park, CA.

Gu, T., Pung, H. K. and Zhang, D. Q. (2005), 'A service-oriented middleware for building context-aware services', *Journal of Network and Computer Applications (JNCA)* **28(1)**, 1–18.

Iwasaki, Y. (1997), 'Real-world applications of qualitative reasoning', *IEEE Intelligent Systems* **12**, 16–21.

JBoss (2010), 'Drools - business logic integration platform', http://www.jboss.org/drools/.

Jena 2 Semantic Web Toolkit (2010), http://jena.sourceforge.net/.

Jess (2008), 'Jess, the rule engine for the java platform', http://www.jessrules.com/.

Ko, E. J., Lee, H. J. and Lee, J. W. (2007), 'Ontology-based context modeling and reasoning for u-healthcare', *IEICE - Trans. Inf. Syst.* **E90-D**(8), 1262–1270.

Kofod-Petersen, A. and Aamodt, A. (2006), Contextualised ambient intelligence through case-based reasoning, *in* 'ECCBR', pp. 211–225.

Kumar, K. A., Singh, Y. and Sanyal, S. (2009), 'Hybrid approach using case-based reasoning and rule-based reasoning for domain independent clinical decision support in icu', *Expert Syst. Appl.* **36**(1), 65–71.

Leake, D. B. (1996), *Case-Based Reasoning: Experiences, Lessons and Future Directions*, MIT Press, Cambridge, MA, USA.

Leake, D. B., Maguitman, A. G. and Reichherzer, T. (2006), Cases, context, and comfort: Opportunities for case-based reasoning in smart homes., *in* J. C. Augusto and C. D. Nugent, eds, 'Designing Smart Homes', Vol. 4008 of *Lecture Notes in Computer Science*, Springer, pp. 109–131.

Magnani, L. and Nersessian, N., eds (2002), *Model Based Reasoning*, Kluwer Academic Publishers.

Mntyjrvi, J. and Seppnen, T. (2003), 'Adapting applications in handheld devices using fuzzy context information', *Interacting with Computers* **15**(4), 521 – 538.

Mylonas, P., Vallet, D., Castells, P., FernÁndez, M. and Avrithis, Y. (2008), 'Personalized infor-

mation retrieval based on context and ontological knowledge', *Knowl. Eng. Rev.* **23**(1), 73–100.

Novk, V., Perfilieva, I. and Mockor, J. (1999), *Mathematical Principles of Fuzzy Logic*, Springer.

Ongenae, F., Strobbe, M., Hollez, J., Jans, G. D., Turck, F. D., Dhaene, T., Demeester, P. and Verhoeve, P. (2008), 'Design of a semantic person-oriented nurse call management system', *Int. J. Web Grid Serv.* **4**(3), 267–283.

Prentzas, J. and Hatzilygeroudis, I. (2007), 'Categorizing approaches combining rule-based and case-based reasoning', *Expert Systems* **24**(2), 97–122.

Racer (2010), 'Racerpro: Renamed abox and concept expression reasoner.', http://www.racer-systems.com/.

rong Jih, W., jen Hsu, J. Y., chang Lee, T. and lu Chen, L. (2008), 'A multi-agent context-aware service platform in a smart space', *Journal of Computers* .

Sowunmi, F., Olaleye, O., Abiola, O., Salako, M. and Eleyoowo, I. (2009), 'A markov chain approach to the dynamics of vehicular traffic characteristics in abeokuta metropolis', *Research Journal of Applied Sciences, Engineering and Technology* **1**, 160–166.

Stamoulakatos, T. S., Kyriazakos, S., Sykas, E. D. and Street, H. P. (2007), 'Hidden markov modeling and macroscopic traffic filtering supporting location based service', *Wireless Communications & Mobile Computing* **7**.

Strobbe, M., De Jans, G., Hollez, J., Goeminne, N., Dhoedt, B., De Turck, F., Demeester, P., Pollet, T. and Janssens, N. (2006), Design of an open context-aware platform enabling desk sharing office services, *in* 'Proceedings (on CD-ROM) of PSC2006, the 2006 International Conference on Pervasive Systems & Computing (part of the 2006 World Congress in Computer Science)'.

Strobbe, M., Hollez, J., De Jans, G., Van Laere, O., Nelis, J., De Turck, F., Dhoedt, B., Demeester, P., Janssens, N. and Pollet, T. (2007), Design of casp: an open enabling platform for context aware office and city services, *in* 'Proceedings of MUCS 2007, 4th International Workshop on Managing Ubiquitous Communications and Services'.

Strobbe, M., Van Laere, O., Dauwe, S., Dhoedt, B., De Turck, F., Demeester, P., van Nimwegen, C. and Vanattenhoven, J. (2010), 'Interest based selection of user generated content for rich communication services', *J. Netw. Comput. Appl.* **33**(2), 84–97.

The Open Services Gateway Initiative (OSGi) (2011), 'The Open Services Gateway Initiative (OSGi)',http://www.osgi.org.

W3C (2004), 'OWL Web Ontology Language Overview', http://www.w3.org/TR/owl-features.

Author Biographies

Matthias Strobbe received his M. Sc. degree in Computer Science Engineering from Ghent University, in July 2004. Since August 2004 he is affiliated with the Department of Information Technology of Ghent University, first as a research engineer and as of October 2006 as a Ph.D. student. The topic of his research covers software architectures and algorithms for support of context awareness and location based services. During his Ph.D. research he was funded by the F.W.O.-V., the Fund for Scientific Research Flanders. While finishing his Ph.D. thesis, in April 2011 he started working on smart energy grids, both as a researcher and project manager.

Olivier Van Laere Olivier Van Laere received his M. Sc. degree in Computer Science in July 2004, and his M. Sc. degree in Computer Science Engineering in March 2007, both from Ghent University, Belgium. As of October 2006, he is an assistant at the Faculty of Engineering, affiliated with the Department of Information Technology of Ghent University. Apart from educational activities, he is currently doing research for his Ph.D. The topic of his research covers the study and development of architectures and algorithms for management of context information, which can be used in advanced multimedia and mobile services. As of 2009, he is focussing on geographical information retrieval and georeferencing multimedia resources.

Bart Dhoedt received a Masters degree in Electro-technical Engineering (1990) from Ghent University. His research, addressing the use of micro-optics to realize parallel free space optical interconnects, resulted in a Ph.D. degree in 1995. After a 2-year post-doc in opto-electronics, he became Professor at the Department of Information Technology.

Bart Dhoedt is responsible for various courses on algorithms, advanced programming, software development and distributed systems. His research interests include software engineering, distributed systems, mobile and ubiquitous computing, smart clients, middleware, cloud computing and autonomic systems. He is author or co-author of more than 300 publications in international journals or conference proceedings.

Filip De Turck received his M.Sc. degree in Electronic Engineering from the Ghent University, Belgium, in June 1997. In May 2002, he obtained the Ph.D. degree in Electronic Engineering from the same university. During his Ph.D. research he was funded by the F.W.O.-V., the Fund for Scientific Research Flanders. From October 2002 until September 2008, he was a post-doctoral fellow of the F.W.O.-V. and part time professor, affiliated with the Department of Information Technology of the Ghent University. At the moment, he is a full-time professor affiliated with the same department in the area of telecommunication and software engineering. Filip De Turck is author or co-author of more than 300 papers published in international journals or in the proceedings of international conferences. His main research interests include scalable software architectures for telecommunication network and service management, performance evaluation and design of new telecommunication and eHealth services.

Piet Demeester is professor in the faculty of Engineering at Ghent University. He is head of the research group "Intec Broadband Communication Networks" (IBCN) that is part of the Department of Information Technology (INTEC) of Ghent University and that also belongs to the Interdisciplinary Institute for Broadband Technology (IBBT). He is Fellow of the IEEE.

After finishing a PhD on Metal Organic Vapor Phase Epitaxy for photonic devices in 1988, he established a research group in this area. In 1992 he started research on communication networks and established the IBCN research group. The group is focusing on several advanced research topics: Network Modeling, Design & Evaluation; Mobile & Wireless Networking; High Performance Multimedia Processing; Autonomic Computing & Networking; Service Engineering; Content & Search Management and Data Analysis & Machine Learning. The research of IBCN resulted in about 50 PhD's, 800 publications in international journals and conference proceedings, more than 20 international awards and 3 spin-off companies.

Correspondence and offprint requests to: Matthias Strobbe, Ghent University - IBBT, Department of Information Technology, Gaston Crommenlaan 8, bus 201, 9050 Ghent, Belgium. Email: matthias.strobbe@intec.ugent.be